PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry oh High Education and Scientific Research

University of Blida1

Faculty of Sciences

Computer Science Department

By

Mr. BENALIOUAT Ahmed

Thesis for obtaining the Master's degree

Field: Mathematics and Computer Sciences

Specialty: Computer Networks and Information Systems

Theme :

Design and Implementation of a Simulator for Performance Analysis of Queueing Systems with Working Vacation

Members of the thesis committee:

--Dr. M. Benyahia	President	University of Blida 1
--Prof. A. Oueld Aissa	Examinator	University of Blida 1
--Dr. M. MEZZI	Internal Supervisor	University of Blida 1

July 2023

Abstract

Overall, the working vacation model offers a balanced approach that combines the benefits of continuous service, reduced queue overflow, faster recovery, and flexibility in handling workload variations. By striking a balance between service provision and resource management, the model aims to optimize client satisfaction, minimize waiting times, and ensure efficient utilization of server resources.

The research findings from the simulation experiments shed light on the advantages and drawbacks of the working vacation queue system compared to the traditional vacation queue. It is likely that the working vacation system demonstrated improved performance in certain scenarios due to its ability to handle low-intensity work periods effectively.

Keywords: *Working vacation queue, Discrete time simulation, Working vacation policy, Performance evaluation*

الملخص

بشكل عام، يقدم نموذج الإجازة المُعمل به نهجًا متوازنًا يجمع بين فوائد الخدمة المستمرة وتقليل تجاوزات الطابور، وزيادة الاسترداد السريع، والمرونة في التعامل مع تباينات الأعباء العملية. من خلال إيجاد توازن بين توفير الخدمة وإدارة الموارد، يهدف النموذج إلى تحسين رضا العملاء، وتقليل أوقات الانتظار، وضمان استخدام كفاءة لموارد الخادم.

تُسلِّط نتائج البحث الناتجة عن تجارب المحاكاة الضوء على مزايا وعيوب نظام صف الإجازة المعمل به مقارنةً بنظام صف الإجازة التقليدي. من المحتمل أن يكون نظام الإجازة المعمل به قد أظهر أداءً محسَّنًا في سيناريوهات معينة بسبب قدرته على التعامل بفعالية مع فترات العمل ذات الكثافة المنخفضة.

كلمات مفتاحية: طابور بالإجازة المعمل به. محاكاة بالزمن المتماثل الانقطاعي. سياسة الإجازة المعمل به. تقييم الأداء.

Résumé

Dans l'ensemble, le modèle de vacances de travail offre une approche équilibrée qui combine les avantages d'un service continu, d'une réduction du débordement de file d'attente, d'une récupération plus rapide et d'une flexibilité dans la gestion des variations de charge de travail. En établissant un équilibre entre la fourniture de services et la gestion des ressources, le modèle vise à optimiser la satisfaction du client, à réduire les temps d'attente et à assurer une utilisation efficace des ressources du serveur.

Les résultats de recherche issus des expériences de simulation mettent en lumière les avantages et les inconvénients du système de file d'attente de vacances de travail par rapport au système traditionnel de file d'attente de vacances. Il est probable que le système de vacances de travail ait démontré des performances améliorées dans certains scénarios en raison de sa capacité à gérer efficacement les périodes de faible intensité de travail.

Mots clés;

File d'attente de vacances de travail, Simulation en temps discret, Politique de vacances de travail, Évaluation des performances.

Dedications

I dedicate this work to my parents. Their unconditional love, trust in me and constant encouragement have been a constant source of inspiration for me. Their sacrifice and unwavering support have been my motivation in my pursuit of knowledge and academic achievement. I am forever grateful for their belief in my abilities and for always supporting and encouraging me even in the most difficult times.

I would also like to thank my extended family members for their unwavering support. Her encouragement and belief in my potential is what fuels my efforts.

Ahmed

Acknowledgment

I would like to take this opportunity to express my sincere thanks to those who have made significant contributions to the realization of this project. Your unwavering support, encouragement and guidance throughout this journey has been invaluable.

First, I am deeply thankful to Mrs. Mezzi for her instrumental role in making this thesis possible. Her invaluable guidance, mentorship, and constructive feedback have shaped this work and enriched my understanding of the subject matter, with special thanks to Mr. Kamel Mouzaï who played a primary role behind the scenes.

Special thanks go to Mr. Ould Aissa and Mr. Ould Khaoua for their support and encouragement throughout this journey. Their belief in my abilities and kind words of encouragement have bolstered my confidence and determination to see this through to completion.

Lastly, I wish to extend my sincere appreciation to Mme. Boutoumi, who provide me with the thesis idea, and to whom I owe an apology.

To everyone who has been a part of this thesis, whether mentioned here or not, your support has made a significant impact on my academic and personal growth. I am humbled and grateful for the kindness and generosity shown to me.

BENALIOUAT Ahmed

Table of content

General Introduction

1. Introduction 3
2. Problematic 3
3. Objectives 4
4. Thesis organization 4

Chapter I: State of the art

1. Introduction 6
2. Performance evaluation 6
3. Simulation 7
   - 3.1. Definition 8
   - 3.3. Collection simulation data 10
     - **3.3.1. The subinterval method** 10
     - **3.3.3. The Replication Method** 11
   - 3.4. Discussion 12
4. Queuing theory 12
   - 4.1. Classic Queue 13
     - **4.1.1. Kendall's notation** 13
     - **4.1.2. Arrival and Service** 14
     - **4.1.3. Discrete-time Markov chain** 15
     - **4.1.4. Random variable:** 16
     - **4.1.4. Exponential law** 17
     - **4.1.5. Performance measures of a queueing system [13]** 17
   - 4.2. Working vacation queue 18
     - **4.2.1. Vacation Models** 19

3.3.2 Vacation policy 19

5. Some related work 20

6. Conclusion 20

Chapter II: Solution modelling

1. Introduction 21

2. Model overview 21

3. Queue detailed description 22

4. Expected advantage of the system 26

Chapter III: Implementation and tests

1. Introduction 29

2. Working environments 29

2.1. Hardware Environment 29

2.2. Software environment 29

3. The tool overview 30

3.1. Arrival process 31

3.2. Service time 31

3.3. Application output 32

4. Testing 33

4.1. Policy 33

4.2. Results 34

4.3. Long simulation run 35

4.4. The Subinterval method 36

4.5. The regenerative method 38

4.6. The replication method 40

4.7. Vacation and Working Vacation 43

5. Conclusion 46

1. Discussion 48

2. Outlooks 48

List of figures

Figure 1: Example of continuous simulation. 9

Figure 2: Example of discret simulation 9

Figure 3: Example of the subinterval method 10

Figure 4: Example of regenerative method. 11

Figure 5:Example of the replication method. 12

Figure 6: M/M/1 queue model 14

Figure 7: Diagram for an example of Markov chain 16

Figure 8:model of M/M/1/k queue with vacation 18

Figure 9: The proposed queue model. 21

Figure 10:Queue model in step 1 22

Figure 11: Diagram of transition in step 1 22

Figure 12: Queue gate 23

Figure 13: Diagram of transition during working vacation 23

Figure 14: the loop process of the system 24

Figure 15: Poisson distribution 31

Figure 16: Exponential distribution 32

Figure 17: Queue size during all steps of the simulation. 35

Figure 18: Result of a simulation with working vacation 34

Figure 19: Queue size during all steps of the long simulation run. 36

Figure 20: Average queue size 37

Figure 21:Duration of Working vacation in the subinterval method 37

Figure 23:Subinterval method total client 38

Figure 24: Regenerative method average queue 39

Figure 25: Working vacation in regenerative method 39

Figure 22: working vacation time 40

Figure 26: Replication method queue size 41

Figure 27:Workin vacation duration in Replication method 42

Figure 28: Queue size of regular queue 44

Figure 29: Working vacation model queue size 44

Figure 30: Vacation queue size 45

List of tables

Table 1 Subinterval method result: 38

Table 2:Regenerative method result 40

Table 3: Simulation parameters 41

Table 4: Replication method result 42

Table 5: Result of the simulation of the 3-queueing models 45

List of equations

4-1Interarrival time..... 14

4-2 Service time..... 15

4-3 Markov property 16

4-5 Discrete random variable 17

4-6 Continuous random variable 17

4-7 Exponential distribution..... 17

3-1 Average queue length..... 25

3-2 Average waiting time 25

3-3 Mean service rate 25

3-4 Vacation total duration..... 26

3-5 Average vacation duration 26

3-6 Vacation time percentage..... 26

List of abbreviations

CPU:	Central processing unit
PE:	Performance Evaluation
OOP:	Object Oriented Programming
Ram:	Random access memory
SSD:	Solid-state drive
WV:	Working vacation
FIFO:	First-In, First-Out
SJF:	Short job first

General Introduction

1. Introduction

Waiting in queues seems to be an ordinary phenomenon of our daily lives. Think of the many times you've had waiting in line over the past month or year and the disappointment that has accompanied those times of waiting. Whether we are in line at the checkout in a hypermarket or in the bank, or sometimes waiting for the turn in the doctor's office or in the pharmacy. Nowadays, taking into account the intensity of competition, a customer who stays too long in the queue is potentially a lost customer. Therefore, understanding the nature of queues and learning how to manage them is a very important area.

The concept of queues with classic vacancy in general, the server stops the service additionally during the vacancy period, as it can do additional work or preventive maintenance. However, there are many situations where the server runs at a very slow pace (lower than normal service rate) rather than stopping service completely during the typical holiday. Hence this class of semi-holiday policy was introduced by Servi and Finn in 2002.

This type of vacation is called working vacation or it can be applied in several fields for the evaluation of the performances of the computer networks, the systems of communication, production and services…etc.

2. Problematic

From a technical point of view, the consideration of the phenomenon of waiter vacancies has generated many analytical difficulties. This makes the classical methods and the results obtained in the theory of classical queuing inadequate. This explains the recourse of researchers to approximation and simulation techniques and to numerical algorithms for the development of performance indices in the field of waiting queues with vacancies.

3. Objectives

In this end-of-study project, we propose to develop a simulation tool based on discrete-event simulation techniques in order to evaluate the performance of queuing systems with vacancies. The project will be implemented in several stages. First, a standard simulation model for a Markovian queuing model will be developed. This model will then be extended to working holiday periods that follow exponential distributions, deterministic and general.

4. Thesis organization

To meet the aforementioned objectives, we have organized our thesis as follows:

- Chapter 1: gives a brief overview of the theoretical notions that surrounds our project. We will begin the chapter by defining the simulation and its pipeline. Then we will talk about the queuing theory and finally we will present the working vacation queue concept.
- Chapter 2: is dedicated to our solution modelling. This latter will begin by the model overview, then we will go through a detailed description of each step of the model. Finally, we will present the performance measures that will be used and the expected advantages of the simulation.
- Chapter 3: concerns our simulation and tests. In this chapter, we have focused on presenting the results that have been obtained for our Simulator of Performance Analysis of Queueing Systems with Working Vacation summarizing them in different use case scenarios using charts and tables.

Chapter I:
State of the art

1. Introduction

Performance evaluation (PE) is used for various purposes in different contexts, humans evaluate the performance of various systems to understand how well they are functioning, identify areas for improvement, and make informed decisions based on the results, among the serval method for releasing the PE simulation is definitely one of the most efficient ways.

Simulation is a powerful technique used to evaluate the performance of various systems and processes. It provides a means to model and analyse complex real-world scenarios in a controlled and virtual environment.

Simulation can be applied to a wide range of domains, including manufacturing, transportation, healthcare, logistics, telecommunications, and many others. For example, in manufacturing, simulation can help analyse production lines, optimize resource allocation, and minimize bottlenecks. In healthcare, simulation can be used to evaluate patient flow, assess the impact of different scheduling policies, or simulate the effects of capacity changes on waiting times.

In this chapter, we will give a brief overview of the concepts related to our research subject.

2. Performance evaluation

The performance evaluation process is used to measure or estimate the behaviour of the system during the implementation phase of its model. Each valid model that is built or developed can be effectively used for predicting system performance in planning, design, or operational steps. One approach to performance measurement is to obtain data by observing events and activities in an existing system. Performance modelling involves representing the system with a model and manipulating the model to gather information about its behaviour and performance [4].

It can be done using methods such [14-16]:

- **Simulation:** the method will be focusing on in this thesis.

- **Surveys and questionnaires:** used to describe the set of questions an individual is being asked. A survey is the process of collecting, analysing and interpreting data from many individuals.
- **Observations and monitoring:** Most evaluation teams conduct some fieldwork, observing what is actually going on at assistance activity sites. Often, this is done informally, without much thought to the quality of data collection. Direct observation techniques allow for a more systematic, structured process, using well-designed observation record forms.
- **Comparative analysis:** the process of comparing items to one another and distinguishing their similarities and differences.
- **Data analysis:** used to provide a deep understanding into the performance of a solution in comparison to the value it brings.
- **Benchmarking:** Benchmark testing compares performance testing results against performance metrics that are agreed upon in the organization based on different industry standards. It helps determine the quality standards of every software application that belong to an organization. Benchmark testing covers software, hardware, and network performance. The goal for benchmark testing is to test all the current and future releases of an application to maintain high-quality standards.

3. Simulation

Traditional discrete-event simulations employ an inherently sequential algorithm. In practice, simulations of large systems are limited by this sequentially, because only a modest number of events can be simulated. Distributed discrete-event simulation (carried out on a network of processors with asynchronous message-communicating capabilities) is proposed as an alternative; it may provide better performance by partitioning the simulation among the component processors. The basic distributed simulation scheme, which uses time encoding, is described. Its major shortcoming is a possibility of deadlock [18]. Several techniques for deadlock avoidance and deadlock detection are suggested. The focus of this work is on the theory of distributed discrete-event simulation.

3.1. Definition

It is the imitation of the operation of a real-world process or system over time, computer-based simulation can be used to imitate the behaviour of the system over time.

From the simulation, data are collected and used to estimate the system performance measures [5].

Simulation can be used as a pedagogical device to reinforce analytic solution methodologies. It can be used to experiment with new designs or policies before implementation and used to verify analytic solutions.

The modem system (factory, wafer fabrication plant, service organization, etc.) is so complex that its internal interactions can be treated only through simulation [4].

The general pipeline of a simulation can be broken up to the following steps:

- Problem formulation.
- Setting of objectives and overall project plan.
- Model conceptualisation.
- Data collection.
- Model translation.
- Verification
- Validation
- Experimental design.
- Production runs and analysis.
- More Runs.
- Documentation and reporting.
- Implementation

3.2. Simulation types

There are various ways to classify simulation depending on the system to be studied, and because this study is about simulation queue in discreet time, we can categorize the simulation in two types [6]:

- **Static simulation:** This type of simulations is often called as Monte Carlo simulation which is a type of simulation that relies on repeated random sampling and statistical analysis to compute the results.
- **Dynamic simulation:** which can further be categorized to |7]:

- **Continuous event simulation:** a simulation in which entities represent a continuously changing quantities like temperature and time are continuous within a defined range. generally used to simulate electrical circuits, chemical reactions …

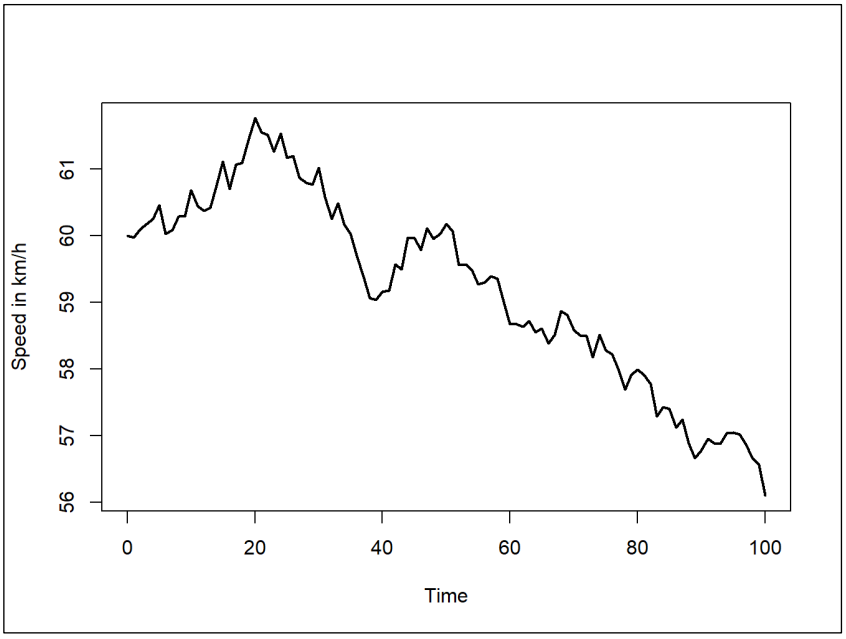

Figure 1: Example of continuous simulation. [6]

- **Discrete event simulation:** a simulation in which entities are considered separate such as clients, servers … the time in it represented as sequence of steps or events that declare a change of state like arrival and departure**,** used perfectly to simulate queuing system

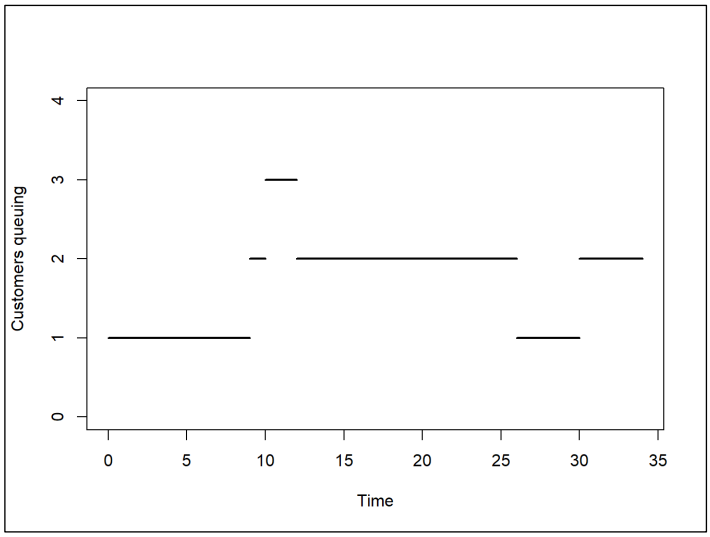

Figure 2: Example of discret simulation [6]

3.3. Collection simulation data

3.3.1. The subinterval method

This method involves a single long simulation run, divided into an initial transient period and several batches. Each batch is treated as a separate simulation run, while no observations are made during the transient period (warm-up interval). The transient period is considered a warm-up and is excluded from the observations. The figure in the text provides an example of this method.

Typically, the batch intervals are chosen to be of equal size, although it is not mandatory. If there is a reasonable logic behind it, batch intervals of different lengths can be chosen.

Selecting a large batch interval size ensures that the batches are independent, allowing for accurate confidence estimation. One advantage of this method is that only one transient (warm-up) interval needs to be taken into account and excluded when recording observations. Since the simulation runs for a long time, any negative effects caused by not properly excluding the transient period during the warm-up interval are minimized [17].

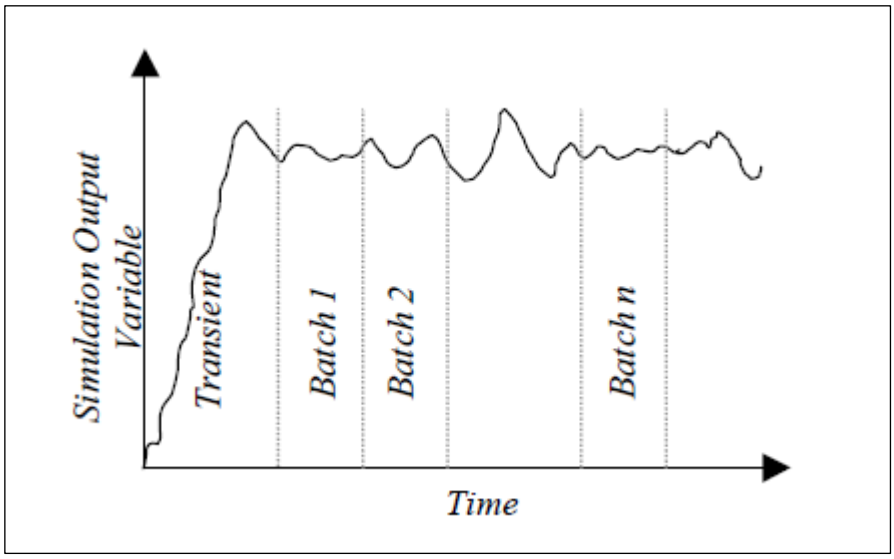

Figure 3: Example of the subinterval method [17]

3.3.2. The regenerative method

This method aims to address the issue of correlation between batches that can arise in the Subinterval Method. Similar to the previous method, we use a long simulation run, but in this case, we identify a specific state of the system known as the regenerative state. The regenerative points are the time instances when the system reaches this state. Batches start and end at these regenerative points once the system has reached a steady state. Figure 4 provides an illustration of this method.

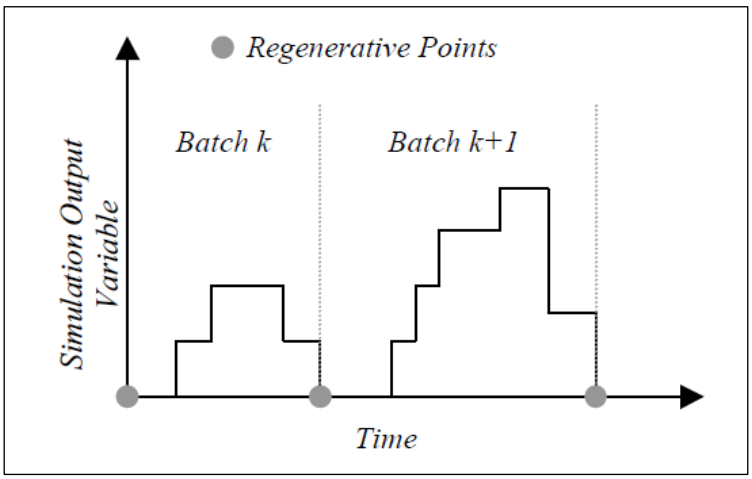

Figure 4: Example of regenerative method. [17]

This method is qualitatively better at ensuring independence between successive batches than the earlier Subinterval Method. It does however have few problems such as [17]:

- Difficulty in identifying the regenerative state.
- Limited applicability to certain simulation scenarios.
- Complexity in handling transient periods and determining steady state.
- Residual correlation between batches despite attempts to reduce it.
- Assumption of stationarity may not hold in all cases.

3.3.3. The Replication Method

This method is the one most popularly used. This was basically the suggested technique for getting ***n*** independent runs of the simulation experiment by running the simulator ***n*** times with different initial random seeds for the simulator's random number generator.

Figure 5 bellow, illustrates this method where the independent runs for the two batches, batch ***k*** and batch ***k+1*** are shown with their own transient (warm up) intervals. and the time intervals during which the system reaches steady state in these two runs [17].

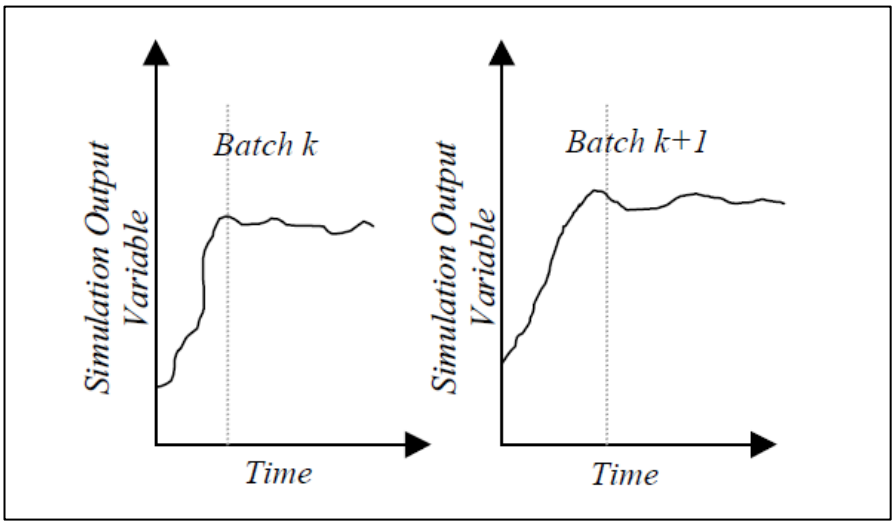

Figure 5:Example of the replication method. [17]

3.4. Discussion

In conclusion, simulation is an invaluable technique for studying complex systems and making informed decisions. It can investigate different scenarios, evaluate key performance indicators and estimate confidence intervals. However, the simulation approach also brings its own challenges. Whether dealing with transition periods, viewing dependencies between batches, or determining appropriate regeneration states, careful consideration and understanding of the specific simulation environment is critical. By using appropriate methods and considering constraints, simulations can provide valuable insights and support the decision-making process.

4. Queuing theory

A couple of clients waiting their turn to see the doctor, some computer instruction is arranged in order to be executed, while both seem to be different process, they both represent a queuing model, whether the system is real or logical human always try to find the optimal way to enhance the performance of these system in order to gain the best income or to reduce the cost or just make the process simple, these effort lead eventually to creating of the queuing theory.

In early 20th a Danish mathematician and engineer called Anger Krarup Erlang (1878 - 1929) used to work for the Copenhagen Telephone Exchange, in order to

optimize the telephonic traffic, he sought to determine how many circuits were needed to provide an acceptable level of telephone service, this ended up by the foundation of the queuing theory field [3].

Later in 1953 David George Kendall (1918 - 2007) set a discipline called Kendall's notation which become a standard system used to describe and classify a queue node.

4.1. Classic Queue

A queue is a system with a single or multiple server, waiting area and clients who are severed according to the discipline of the queue.

Example: people waiting their turn in ATM (Automated Teller Machine) is a queue with one server serves 1 client in time, in FIFO (First In First Out) order with random time of arrival and service and unlimited number of clients.

A queue is characterized by:

- **Interarrival rate**: Usually denoted with λ its rate at which entities arrive at the queue, such as customers or tasks.
- **Service rate**: The rate at which entities are served or processed in the queue, denoted by μ.
- **Number of servers:** the service rate can be affected based if the system has a single or multiple server.
- **Queue length:** The number of entities waiting in the queue at a given point in time.
- **Queue capacity**: The maximum number of entities that the queue can hold at a given time. If the queue reaches its capacity, new arrivals may be rejected or have to wait in a separate overflow queue.
- **Queue discipline**: The set of rules determining the order in which entities are served from the queue. such is FIFO (First-in, first-out) SJF (Short job first) or by priorities.

4.1.1. Kendall's notation

Kendall's notation is used to characterize queueing systems. It can be described in this form A/B/C/D/E, where [1]:

- A indicates the arrival time distribution,
- B indicates the service time distribution,
- C indicates the number of servers,
- D indicates the system capacity,
- E indicates the queue scheduling discipline.

In some studies, it is possible to add the size of the customer population.

Some common distributions for A and/or B are M for Markovian, E for Erlang while G is for General cases, since this thesis focus on Markovian distribution, we will use M for arrival and service time.

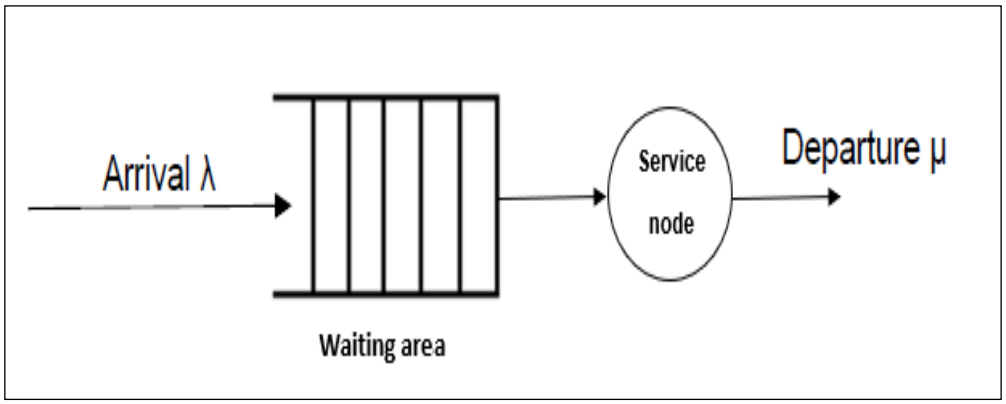

Figure 6: M/M/1 queue model

4.1.2. Arrival and Service

The customer arrival process may be described by the number of arrivals per unit time (the arrival rate) or the time between successive arrivals (the interarrival time) [1].

We use the variable λ to denote the mean arrival rate. In this case, 1/λ denotes the mean time between arrivals. If the arrival pattern is not deterministic, the input process is a stochastic process, which means that we need its associated probability distribution. The probability distribution of the interarrival time of customers is denoted by $A(t)$ where:

$A(t)$ = Prob {time between arrivals ≤ t}

and

$$\frac{1}{\lambda}\int_0^{\infty} t\, dA(t) \qquad 4\text{-}1$$

Interarrival time [1}

where $dA(t)$ is the probability that the interarrival time is between t and $t + dt$.

Same apply for service process where μ denote the mean service rate, and hence $1/\mu$ denotes the mean service time.

$B(x)$ = Prob {Service time ≤ x}

and

$$\frac{1}{\mu}\int_0^{\infty} x\, dB(x) \qquad \textit{4-2 Service time [1]}$$

where $dB(x)$ is the probability that the service time is between x and $x + dx$.

4.1.3. Discrete-time Markov chain

Bibliography: Andrei Andreyevich Markov (1856-1922) was a Russian mathematician who is best known for his work in probability and for stochastic processes especially Markov chains [3].

Definition: A Markov chain is a stochastic process introduced by Andrei Andreyevich Markov in 1907. It is widely used for modeling discrete-time systems where the future state depends solely on the current state, exhibiting the Markov property. The Markov property states that the future state is independent of the past states, given the present state. This property allows for efficient modeling and analysis of systems by focusing on the current state without considering the entire history. In a Markov chain, the sum of transition probabilities of outgoing arrows from any state always adds up to 1, ensuring comprehensive coverage of possible outcomes. Markov chains find diverse applications across various fields due to their ability to capture probabilistic transitions in a wide range of processes [12].

It can be represented by this mathematic formula:

$P(X_{t+1} = x \mid X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0) = P(X_{t+1} = s \mid X_t = x_t)$ 4-3 *Markov property [12]*

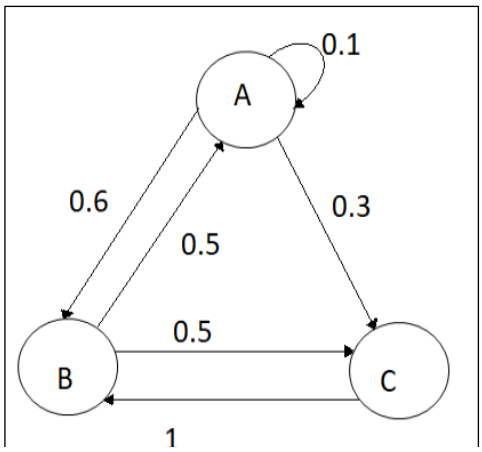

Figure 7: Diagram for an example of Markov chain

For a discrete-time Markov chain, we observe its state at a discrete, but infinite, set of times. Transitions from one state to another can only take place, or fail to take place, at these, somewhat abstract, time instants—time instants that are mostly taken to be one time unit apart. Therefore, we may represent, without loss of generality, the discrete index set T of the underlying stochastic process by the set of natural numbers {0, 1, 2, …}. The successive observations define the random variables *X0, X1, ..., Xn,...* at time steps *0, 1, ..., n, ...,* respectively. Formally, a discrete-time Markov chain {Xn, n = 0, 1, 2, . . .} is a stochastic process that satisfies the following relationship called the Markov property [1]:

For all natural numbers n and all states xn,

$Prob\{X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_0 = x_0\}$

$= Prob\{X_{n+1} = x_{n+1} | X_n = x_n\}.$

4.1.4. Random variable:

A random variable is an application X of a probability space Ω in a state space E which is defined as a set of values that the random variable X could take. [10]

$$X: \Omega \rightarrow E \rightarrow X(\omega)$$

There are two types of random variables:

When E ⊂ Z, X is said to be a discrete random variable which is defined by its state probabilities:p(n) = P[X=n], n = -∞, …, +∞, with:

$$\sum_{n=-\infty}^{+\infty} p(n) = 1$$

4-4 Discrete random variable [10]

When E ⊂ R, X is said to be a continuous random variable which is defined by its probability density function $fx(x)$ for x ∈] − ∞, +∞ [, such that:

$$\int_{-\infty}^{+\infty} fx(x)dx = 1$$

4-5 Continuous random variable [10]

4.1.4. Exponential law

The exponential law with parameter λ ≥ 0, is a continuous-time random variable, whose probability density is defined by:

$$F(t)=\begin{cases}\lambda e^{-\lambda t} & \text{if } t\geq 0 \\ 0 & else\end{cases}$$

Its distribution function is given by:

$$F(t) = \int_{0}^{-t} \lambda e^{-\lambda x} dx = 1 - e^{-\lambda t}, t \geq 0$$

4-6 Exponential distribution

while its average is equal to $\frac{1}{\lambda}$

4.1.5. Performance measures of a queueing system [13]

The following performance measures are usually used to evaluate queuing systems [13]:

- **Traffic intensity ρ:** It's the relation between the arrival and service rate. We have:
    - λ: arrival rate
    - μ: service rate

- $\rho = \frac{\lambda}{\mu}$

- **Server utilization u :** Given by the relation between arrival rate and service rate where m is servers count.

$$u = \frac{\lambda}{m\mu}$$

- **System throughput:** The throughput of a queuing system is the average number of customers that are processed per unit of time. In a queuing system in which all arriving customers are eventually served and leaving the system, the throughput is equal to the arrival rate.
- **Response time T:** The time a client spends in the system from the moment it arrives at the queue until the moment it leaves the server is called response time or sojourn time.
- **Waiting time W:** Waiting time is the time a customer spends in a queue to be processed. Therefore, we have:

Response time = waiting time + service time

- **Average number of customers waiting :** The average number of customers waiting in the queue to get the service is called the average queue length and is noted L_q.

4.2. Working vacation queue

A vacation queue is a queue in which a server takes vacation from a primary task to perform one or more secondary tasks or to rest or to perform some preventative maintenance tasks. During this time, customers are waiting to be served. Such queues can exist in multiple applications [2].

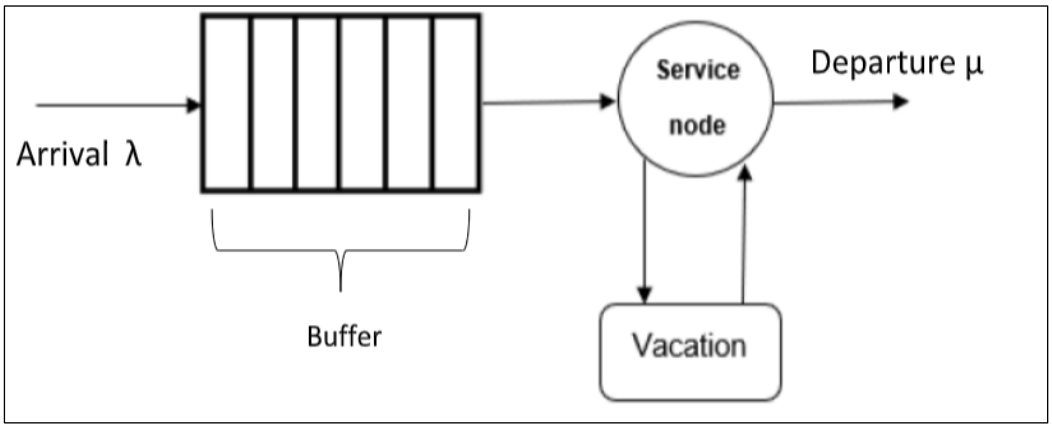

Figure 8:model of M/M/1/k queue with vacation

In working vacation however, the server can be utilized in different rate during the vacation period [19]

4.2.1. Vacation Models

In [9] Rehab F. Khalaf mentioned different types of vacation models:

- **The single vacation model:** In this model, a single vacation is scheduled after the conclusion of each busy period. Once the server returns from this vacation, it does not go on another vacation, even if the system remains empty. This particular vacation scenario can occur in situations such as maintenance activities in production systems, where the maintenance itself can be regarded as a form of vacation.
- **The multiple vacation model:** this type of vacation may arise from cases like maintenance in computer and communication systems where processors in a computer and communication systems do considerable testing and maintenance besides doing their primary functions (processing telephone calls, receiving and transmitting data, etc.). The maintenance work required is divided into short segments. Whenever the customers are absent, the processor does a segment of the maintenance work. When the system is idle, the server takes a vacation (works on a maintenance segment). Upon return from a vacation, the server starts service only if it finds K or more customers waiting in the queue, if the number waiting is less than K then it goes on another vacation (maintenance segment) [9].
- **The limited-service vacation:** in which the server takes a vacation on becoming idle or after having served *m* consecutive customers, or after time *T*.

3.3.2 Vacation policy

A vacation is characterized by Starting Policy, Ending Policy and Vacation Period

- **Starting Policy:** It is the vacation start rule that determines when the server begins a vacation. There are three main types [9]:
    - **Gated service:** in this case, as soon as the server returns from the vacation it places a gate behind the last waiting customer. It then begins to serve only customers who are within the gate, based on some rules of how many or for how long it could serve.
    - **Exhaustive service:** In this case, the server serves customers until the system is emptied, then it goes on vacation.
    - **Limited Service:** In this case, a fixed limit of K is placed on the maximum number of clients that can be served before the server goes on vacation. The server also goes on vacation: when the system is empty or when K clients.

- **Ending policy:** This rule determines when the server resumes service. Three vacancy policies are defined below [8]:

  - **N-vacation:** In this policy the server returns from vacation when the queue size is greater than or equal to N.
  - **T-vacation:** In this case, the server is not activated (on vacation) T units of time after the end of a service period. After this vacation (of length T), the server is reactivated to serve clients only if there was at least one client present, otherwise the server takes another vacancy period of length T.
  - **D-vacation:** In this policy the server returns from a vacation when the sum of the service times of the clients in the buffer exceeds the value D.

- **Vacation duration:** The durations of the vacancy depend on the policy followed, they are considered as independent random variables and identically distributed with a certain law of probability, according to the characteristics of the modelled system [20].

5. Some related work

In 2015 Dong-Yuh Yang and Chia-Huang Wu released a paper investigating a *N*-policy M/M/1 queueing system with WV and server breakdowns, as soon as the system becomes empty, the server begins a WV. The server works at a lower service rate rather than completely stopping service during a vacation period [11].

Zeynep Akşin and Ward Whitt published in 2014 a paper that presents a comprehensive analysis of WV queues, focusing on various models and performance measures. It discusses the impact of different vacation policies and provides insights into optimizing the system performance. The study includes theoretical analysis and simulation results for practical applications ["Working Vacation Queues" by Zeynep Akşin and Ward Whitt (2014)].

A Survey by Ger Koole and Sandjai Bhulai in 2008 provided an overview of the literature on WV queues. covering different models, policies, and performance measures, summarizing the state-of-the-art research in the field. alongside with discussing about the mathematical methods used in analyzing working vacation queues and highlights important research directions [Ger Koole, Sandjai Bhulai, Working Vacations in Queueing Systems: A Survey"].

6. Conclusion

In this chapter, we tried to give a succinct overview of the basic notions that are related to our project. In the next chapter, we will present the

modelling of a Simulator for Performance Analysis of Queueing Systems with Working Vacation.

Chapter II:

Solution modelling

1. Introduction

It is known that a WV (Working Vacation) queue fulfils the role of a perfect combination between a vacation queue and a classic queue, by maintaining a balance between serving the clients and performing secondary tasks, it's no surprise then that many researchers proposed servals model of WV queue over the past years.

In this chapter, we will present a WV M/M/c queue with Poisson arrivals and exponential service time, in which the servers go on vacation after serving a given amount of clients. Firstly, there will be an overview of the model illustrated through the pipeline of the system, then a detailed description of every steps the queue goes in, and finishing by listing the performance measure we will be focusing in.

2. Model overview

The queue will have multiple servers that have as main task looking after client service. In addition, the server should handle serval secondary task.

To avoid the queue overflow, the servers will ignore the secondary tasks as long as the arrival rate is still high. Then, as soon as the servers serve a given number of clients that were waiting in a queue, they begin a working vacation process where the drop of the service rate to a specific amount is about 50% which means doubling the service time. The servers eventually end the WV as soon as the queue reach a size of K. Then, the queue returns to the original phase and repeats the same process.

Bellow, is the general process that represents the model described:

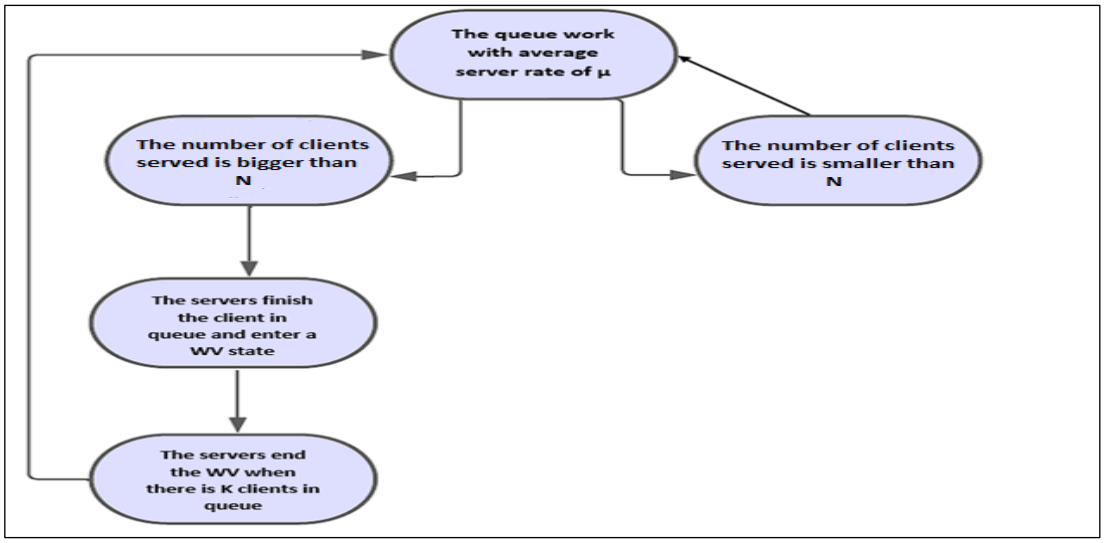

Figure 9: The proposed queue model.

3. Queue detailed description

The queue will be running in an endless loop repeating. In a succession of steps, described thereafter:

We will note the beginning of every loop with time 0.

Step 1: from 0 to n: the head of the loop

The clients start arriving at an average rate of λ represented by Poisson distribution, a c number of servers start serving them with a mean rate of μ, the μ_t will be defined using exponential law.

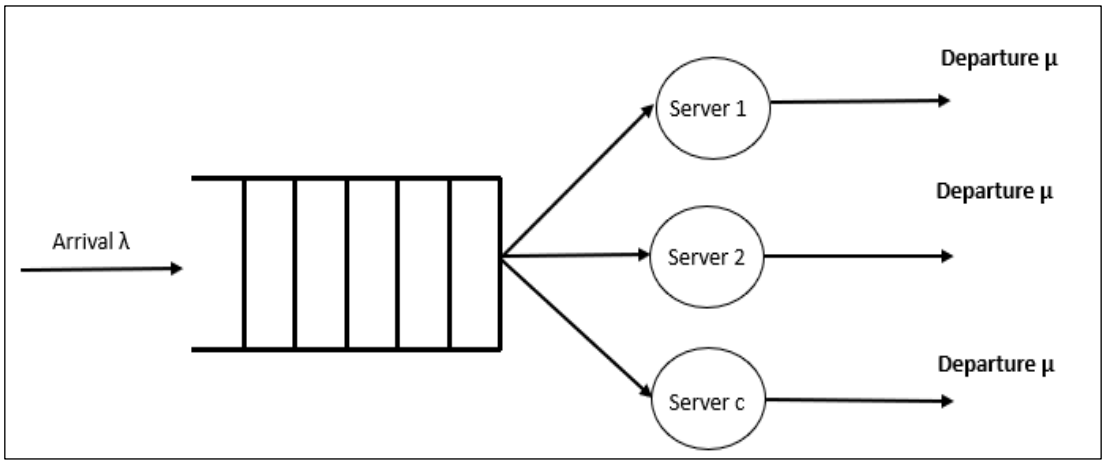

Figure 10:Queue model in step 1

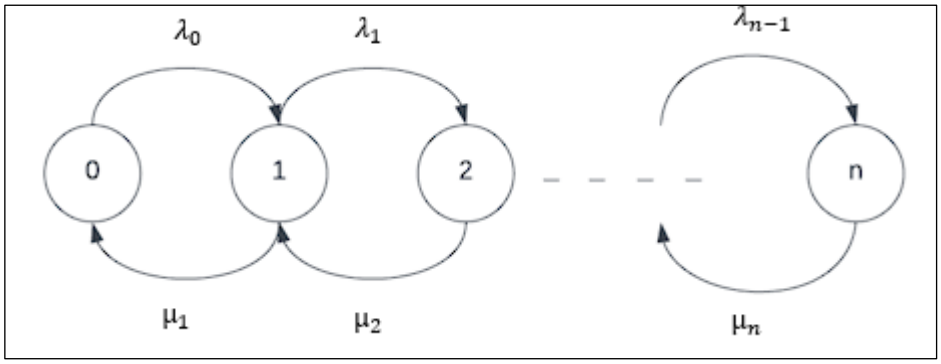

Figure 11: Diagram of transition in step 1

Step 2: At t=n: Preparing for vacation

At n unit of time an N clients have been served, the system responds by placing a gate behind the last client in queue, blocking the arrival and continue to serve the client at the original service rate until the last client before the gate being served.

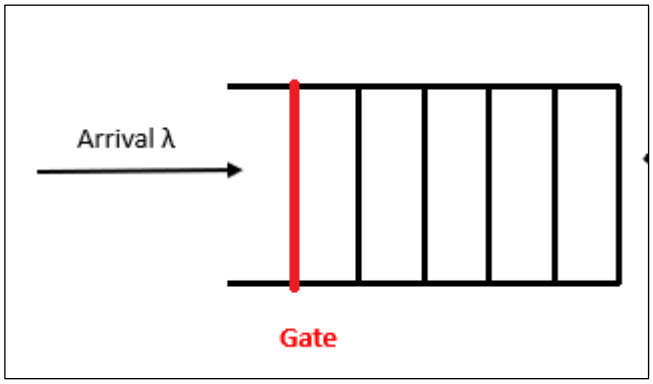

Figure 12: Queue gate

Step 3: Working Vacation

The servers switch to WV state where they begin performing secondary task alongside their primary task which is serving the client, but they lower the service rate by a given amount which is 50% in the figure bellow that's means they system double the service time, the arrival process begins again.

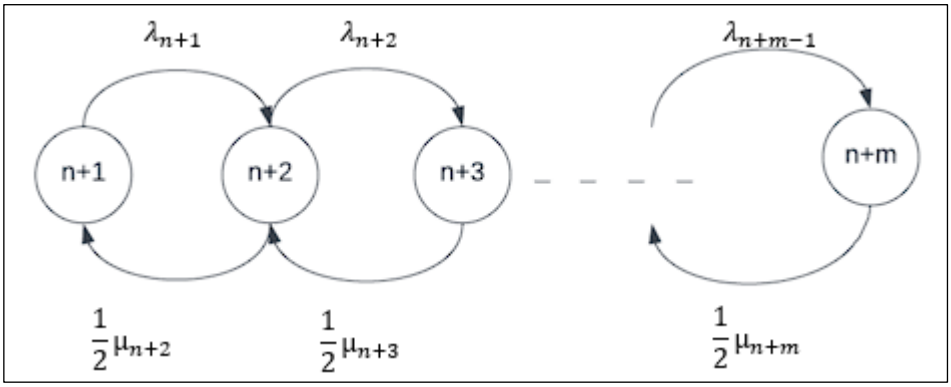

Figure 13: Diagram of transition during working vacation

Step 4: Exiting the WV

At the start of the vacation, the system picks a size k. If the size of the queue reaches K, the severs immediately abandon the secondary task and start again serving the client at the original rate.

Step 5: After the vacation

The system returns to the first state with the original parameters, and the process run in a loop.

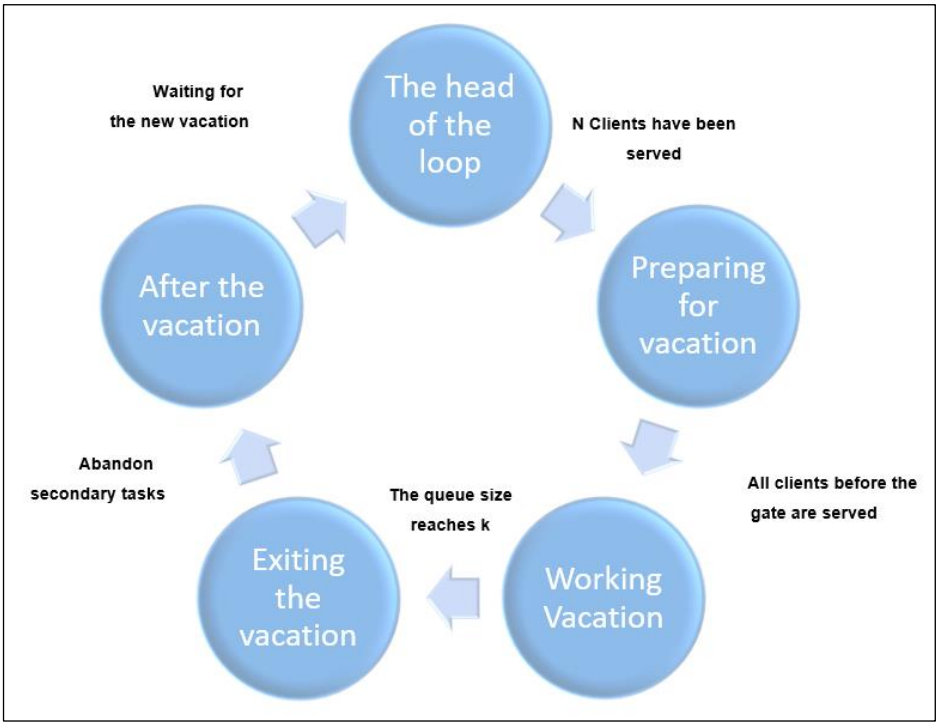

Figure 14: the loop process of the system

4. Performance measures

The main goal of every new model is enhancing the performance of certain system, allowing it to generate the best outcome, decreasing the cost, reducing the use of resources or simply improving the quality of life of the system.

In this model we aim to continually serving the clients while performing secondary system task in order to maintain healthy queue size thus improve the clients' quality of life inside the system, and for that the measure we're interested in are such:

- **Average queue length: L_m**

  Can be a measure of how well the working vacation model handles the workload, obtained by collecting the queue length of every unit of time and divided by the total time.

  It can be described by this formula:

$$\frac{\sum_0^{t_n} L_t}{n+1} \qquad 3\text{-}1$$

Average queue length

$$Where\ L_t\ is\ the\ queue\ size\ in\ time\ t\ and\ n + 1\ is\ the\ total\ time\ of\ Simulation$$

- **Total waiting time: W**

  It is the total time where the queue is not empty, increasing every time there is at least 1 client in queue.

- **Average waiting time: W_m**

  It is the average time a client spends waiting in queue, obtained by dividing the Total waiting time **W** the number of clients C.

$$W_m = \frac{W}{C} \qquad 3\text{-}2$$

Average waiting time

- **Max queue length: L_{max}**

  The peak number of clients waiting in queue

- **Mean service rate: μ_m**

  The average service rate, deducing by dividing the sum of all service rate during the simulation by the total time, given by this formula:

$$\frac{\sum_0^{t_n} \mu_t}{n+1} \qquad 3\text{-}3$$

Mean service rate

- **Vacation total duration: V**

  The sum of the serval WV (V_t) the system passed by

$$V = \sum_{0}^{n} V_t$$

3-4
Vacation total duration

- **Vacation average duration: V_m**
  Total vacation duration divided by the count of vacation N_v

$$V_m \frac{V}{N_v}$$

3-5
Average vacation duration

- **Vacation time percentage:**
  Percentage of vacation time out of total time

$$\frac{v}{n+1} \times 100$$

3-6
Vacation time percentage

This measure will be compared between to another from a vacation queue where the server stops serving the client until a k of client is being served to see how many improvements a WV queue can provide in such a system.

4. Expected advantage of the system

The upside of using a WV model compared to a regular vacation model in this scenario is that it allows the servers to continue providing some level of service even during periods of reduced workload. Here are some advantages of the WV model:

- **Continuous service:** With a WV, the servers continue to handle client service at a reduced rate. This means the servers are still able to process requests and provide some level of service. In contrast, with a regular vacation model, the servers would completely stop working, leading to a complete halt in client service.
- **Reduced queue overflow:** By maintaining a lower service rate during a WV, the servers are able to prevent the queue from overflowing. This helps

in managing client expectations and ensures that the system remains responsive, even during periods of reduced workload. In a regular vacation model, without any service being provided, the queue would continue to grow unchecked, potentially leading to frustration among clients and delays once the vacation ends.

- **Flexibility in workload variations:** The WV model allows for more flexibility in handling workload variations. It accommodates situations where the arrival rate fluctuates, such as during weekends or late hours, by reducing the service rate accordingly. This adaptive approach ensures that resources are optimally utilized and provides a more consistent level of service. In a regular vacation model, the servers are either fully operational or completely inactive, which may not be efficient or responsive to changing workload patterns.

In summary, the WV model has to provide the benefits of providing continuous service at a lower rate, prevents queue overflow, allows faster recovery, and provides flexibility to manage workload fluctuations. These advantages make it a more viable and efficient approach than the scheduled vacation model in situations where a certain level of service needs to be maintained even during periods of reduced workload.

6. Conclusion

In conclusion, the working vacation model for a queue with multiple servers introduces a flexible approach to managing client service during periods of varying workload. By incorporating a working vacation period, where the servers reduce their service rate but continue to provide some level of service, the model offers several advantages compared to a regular vacation model and this what we wanted to implement.

In the next chapter, so, with this chapter ends the theory part and we move on to the experimental part.

Chapter III :
Implementation and tests

1. Introduction

This chapter will focus about the simulation of the Queue that was modelled in the previous chapter. First, will cover both the hardware and software environment of the application, including the language in which the app was programmed, passing to the application overview, the code architecture and finish by small benchmark.

In addition to the working vacation model introduced in chapter 2, the queue will be simulated with a vacation process that follow the same policy of the working vacation model except it stop serving the client completely until the queue reach a k size.

The tool will be focusing on simulation and providing detailed Data about the process in form of txt and xlsx file, with allowing the users to use different queue set up suitable to theirs desire.

2. Working environments

2.1. Hardware Environment

The application was developed with a Desktop computer having the following configuration:

- A processor with 3.9Ghz – 4.4Ghz, 6 cores and 12 threads CPU being the AMD Ryzen 5 5600G
- A RAM of 16 GB 3200Mhz
- and SSD storage.

During an open test the app consume about 29 mb of RAM which is inconsiderable amount according to the modern standards, and surely it took about 12% of CPU usage.

2.2. Software environment

The simulation tool was developed in Microsoft Windows 11, using Microsoft Visual Studio 2022, an integrated development environment (IDE) that allows the users to create deafferent types of projects in serval domains. It provides a comprehensive set of tools and features to assist software developers in writing, testing, and debugging code. It is designed to enhance the productivity of developers by providing a centralized workspace where they can perform various development tasks.

The chosen language for creating the tool was C#, an Object-Oriented Programming language created by Microsoft in early 2000s. C# was chosen for the following reason:

- It is easy to model a real system because the real objects are represented by programming objects in OOP.
- Ideal for developing Windows Desktop apps.
- Has a fast compilation time.
- Highly supported by Microsoft Visual Studio.
- Visual Studio includes built-in testing tools for C# applications.
- It supports various testing frameworks, such as MSTest, NUnit, and xUnit.

3. The tool overview

The tool performs two types of queues, a working vacation queue that was modelled in the previous chapter alongside to a vacation queue which follow the same model, except that the servers go idle in step 3 instead of working at lower rate.

The queue pass by 3 steps during the simulation:

a) **Regular state:**

- This state is executed when stat is equal to 1.
- It starts by checking the queue's size and updating the maximum queue size if necessary (MaxQueue).
- The variable unite is incremented, representing the simulation step.
- Information about the current state, step, and queue size is printed to the console.
- If there are clients in the queue (client.Count != 0), and there is an available server (freeserver(server) != -1), a client is dequeued from the queue and added to the server.
- The server states are printed to the console, and server updates are performed (Updateserver(server)).
- The queue size is printed again, and if needed, a new client is generated, enqueued, and its service time is recorded.

b) Preparation state

- This state is executed when stat is equal to 2.
- It enters a loop that continues until the queue is empty (client.Count > 0).
- The variable unite is incremented, representing the simulation step.
- Information about the current state, step, and queue size is printed to the console.

- The same operations as in the Normal state are performed, including updating the server, queue size, generating new clients, and updating statistics and worksheet.

c) Vacation state

- This state is executed when stat is equal to 3.
- It enters a loop that continues until the queue size reaches a specified value k
  - (client.Count <= k).
- During this period the service rate generator will be multiplied by a given parameter to respect the vacation policy.
- The variable unite is incremented, representing the simulation step.
- Information about the current state, step, and queue size is printed to the console.
- The same operations as in the Normal state are performed, including updating the server, queue size, generating new clients (with double service time), and updating statistics and worksheet.
- After the simulation loop ends, the code may continue with further calculations or analysis using the gathered statistics.

3.1. Arrival process

As was cited in the modelling chapter, the arrival policy follows a Stochastic process, the tool use either Poisson distribution or a Randomizer with a given average arrival rate.

```
Random rnd = new Random();
double lambda = arriverate; // Arrival rate parameter
double L = Math.Exp(-lambda);
double p = 1.0;
int k = 0;

do
{
    k++;
    double u = rnd.NextDouble();
    p *= u;
} while (p > L);

return k - 1;
```

Figure 15: Poisson distribution

3.2. Service time

Follow the exponential policy starting by a given average service time.

- The time could not take less than 2 units and more than 8 units so a “do while” loop is added to insure underwhelming service or the opposite.
- Since the simulation time is measured by steps, the service time should be converted to integer by adding a cast when returning the value.

```
public int GenerateService(int service)
{
    Random random = new Random();
    double u,x;


    do
    {
        u = random.NextDouble();
        x = -service * Math.Log(1 - u);
        x = Math.Round(x);
    } while (x > 8 || x < 2);

    return (int)x;
}
```

Figure 16: Exponential distribution

3.3. Application output

The application generates an xlsx present the simulation result of each unit of time, containing queue size, clients served and the state of the queue. It generates also 2 text files: one shows the final result while the other one describes every step with its detailed result. Including the performance measures discussed in the preview chapter:

- **Average queue size:**

```
While(Unit<=Duration)
{TotQueue ← TotQueue + Client.count; } // in every state
AverageQueue ← TotQueue/totale; //totale (totale Simulation duration)
```

- **Max queue size:**

```
MaxQueue ← 0;
While(Unit<=Duration){
if ( MaxQueue <= Client.count )
MaxQueue ← Client.count
}
```

- **Total clients:**

```
TotalClient ← 0;
While (Unit <=Duration){
If (UpdateQueue){
TotalClient ← TotalClient +1; }
```

- **Total waiting time:**

```
TotalWating = TotalQueue
```

- **Average waiting time:**

```
AvgWaiting ← TotalQueue/TotalClient;
```

- **Average service time:**

```
If (UpdateQueue){
ServiceTime ← GenerateService() +ServiceTime;}
AvgService ←ServiceTime/TotalClient;
```

- **Total Vacation time:**

```
V ← 0
if (stat ==3){
    V ← V+1;}
```

- **Vacation ratio:**

```
Vpercent ← (V/Total) * 100;
```

4. Testing

4.1. Policy

To test the queue will follow the following policy:

- The initial duration of the simulation will be set at 100 units.
- The arrival rate will be generated by randomizer with 0.5 average.
- The service time will be 5 unit in average.
- There will be 3 servers.

- The system prepares for a vacation as soon as 30 clients have been served.
- The servers work during vacation at 50% of their speed.
- The system ends the vacation as soon as the queue is bigger than 5.

4.2. Results

The simulation extended for 40 more unit for a total of 240 (The simulation while in WV or preparing for it).

The system was in simulation at the end from step 169. From then, the queue didn't reach the limit until step 240.

A total of 3 occurrences of each stats happened, the WV took the most spot with 54% of the simulation time, with an average duration of 43 units.

The queue reached its peak in step 90 with 12 clients waiting where its average size was 3.675, while the average service time was 9 units due the domination of the vacation period. Figure 17 presents a summary of the obtained results:

In the other hand the result provided by the xlsx file allows to transform the data into charts for certain measure. The figure 18 represents the queue size during all steps of simulation, it clearly visual that the queue size ascends before quickly drops down, this process keeps going on cycles during the simulation.

```
Total time :                              240

Total queue size :                        882
Average queue size :                        3.675
Max queue size :                          12

Total clients :                           98
The Average waiting time :                 9
Average service rate :                     6.96

Time in normal queue :                    82
Preparing for vacation :                  27
Time in Vacation 131
Occurrence of normal queue stat :          3
Occurrence of preparing for vacation stat : 3
Occurrence of vacation stat :              3
Single Vacation duration :                43.66
Vacation time percentage :                54.58
```

Figure 17: Result of a simulation with working vacation

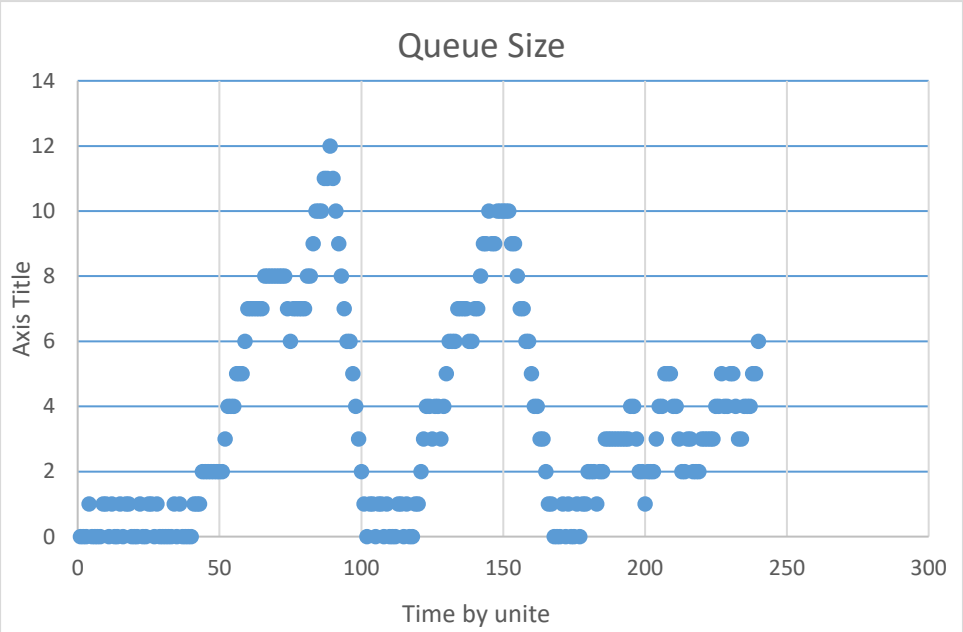

Figure 18: Queue size during all steps of the simulation.

4.3. Long simulation run

This simulation uses the following parameters:

- Random arrival with 0.5 average.
- Service rate follow exponential distribution with average of 5 units.
- There are 3 servers.
- The simulation lasts for 1000 units.
- The system starts preparing for vacation once the 10 clients were served.
- During the working vacation the servers work at double rate.
- The working vacation ends once the queue size surpasses 5 (K=5).

After 1000 units of time the system pass by 16 cycles (a cycle starts from the beginning of Normal phase until the ending of the working vacation phase passing by the preparation phase), the working vacation phase covers 45% from total duration, allowing the servers to have decent time to performs the secondary time. The figure 19 bellow represents the queue size evolution in this scenario.

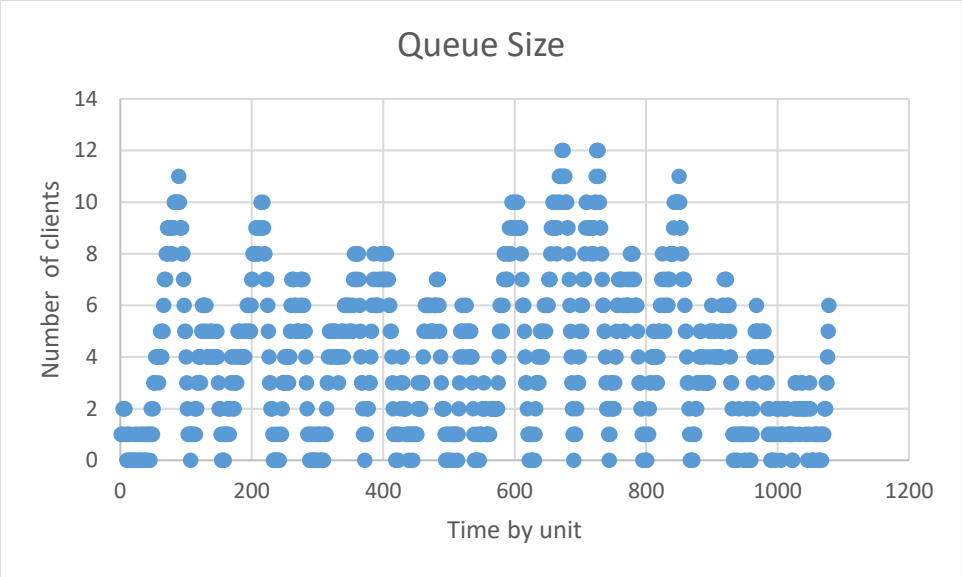

Figure 19: Queue size during all steps of the long simulation run.

4.4. The Subinterval method

The approach of this method estimates performance measures by averaging statistics collected over multiple subintervals. To achieve this, simulation will be running for a long duration that will be divided in 5 intervals.

Applying this method in previous simulation give the results depicted in the figure 20 which represents the average queue length in every period of time:

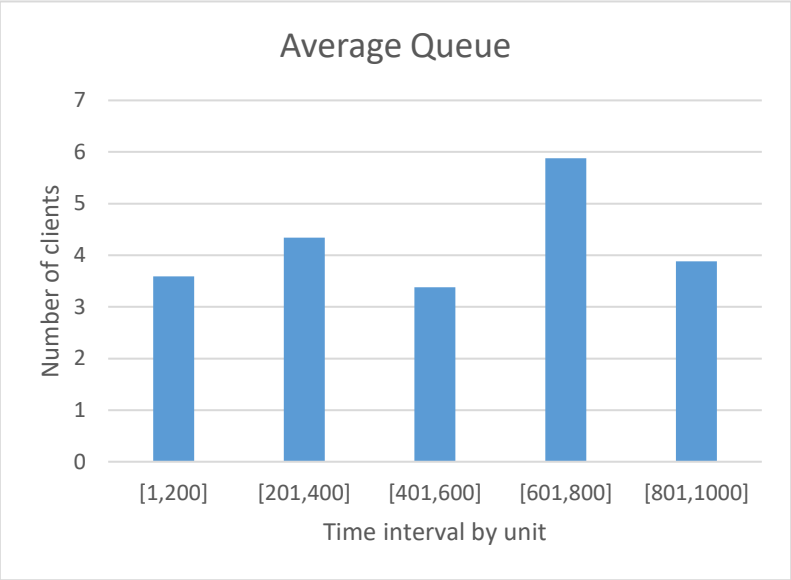

Figure 20: Average queue size

The average in each period indicate that the system's queue length is not stabilizing, it clearly shows that the system did not reach the steady state which is predictable due the stochastic policy that the arrival and service time follow.

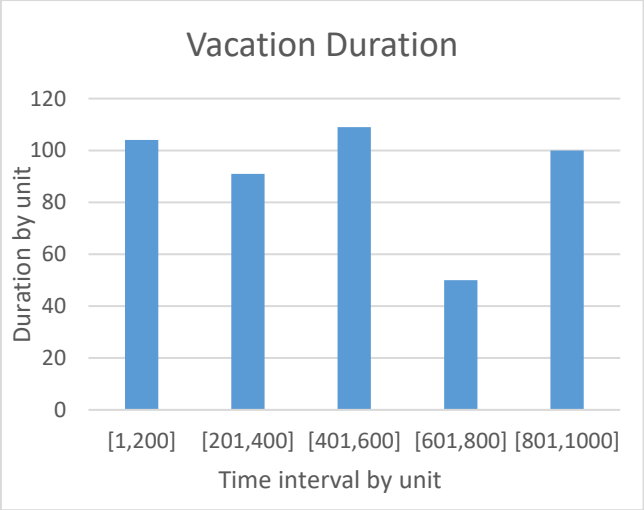

Figure 21:Duration of Working vacation in the subinterval method

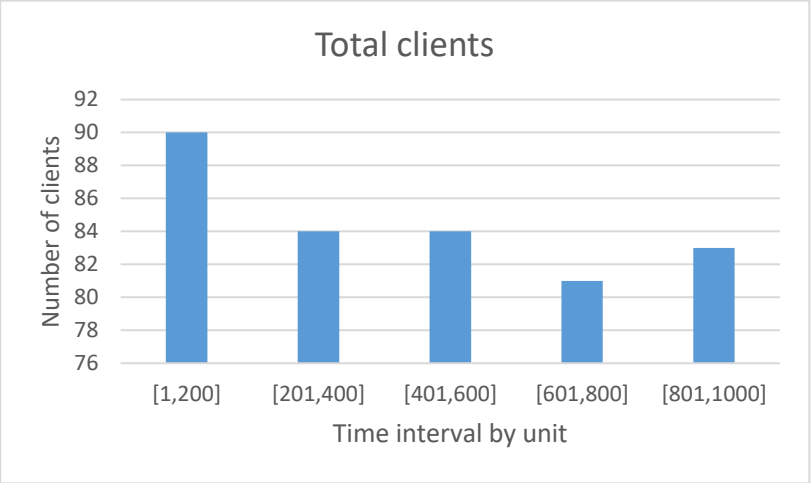

Figure 22:Subinterval method total client

However, the working vacation phase was a bit constant excluding the 4th part, and it was covering almost half of the duration.

The table below summarizes all the results set.

Table 1 Subinterval method result:

Interval	Max Queue	Average Queue	Total waiting	Vacation Duration	Vacation ratio	Total clients
[1,200]	11	3.59	718	104	52%	90
[201,400]	10	4.34	868	91	46%	84
[401,600]	10	3.38	676	109	55%	84
[601,800]	12	5.88	1176	50	25%	81
[801,1000]	11	3.885	777	100	50%	83

4.5. The regenerative method

This method estimates performance measures in systems that exhibit regenerative or cyclic behaviour, making it ideal for this queue system since it run in cycles.

For this method, we run a new simulation without changing the parameters, then divide the result into 5 cycles each cycle begins from the start of normal phase until the end of the working vacation phase, that comes up with the following:

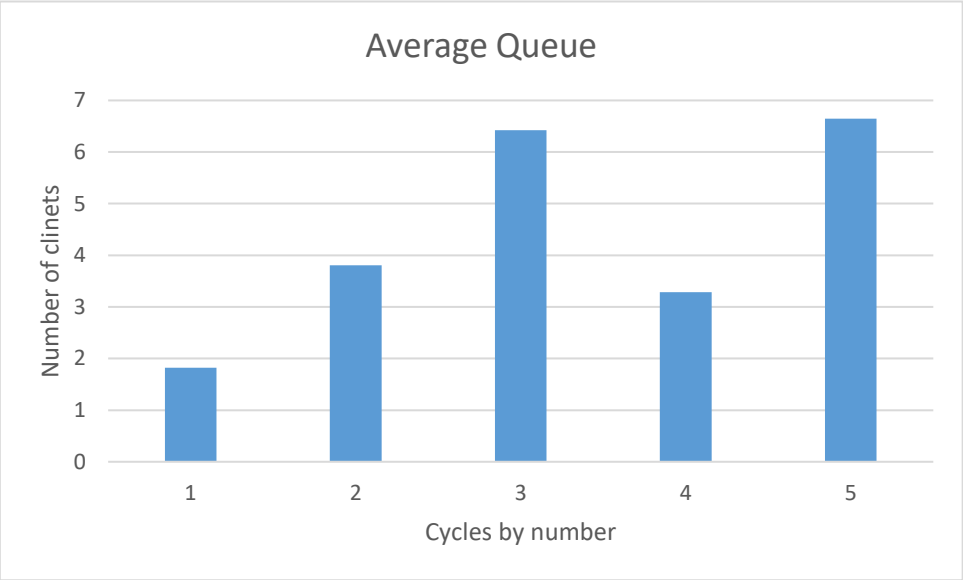

Figure 23: Regenerative method average queue

Similar to the preview simulation and due to the same reasons the queue system will not reach stabilized stat, confirmed by the Working vacation ratio that show huge drop between cycle 2 with 67% and cycle 3 with only 19%. (The statics are in percentage due the duration difference for each cycle)

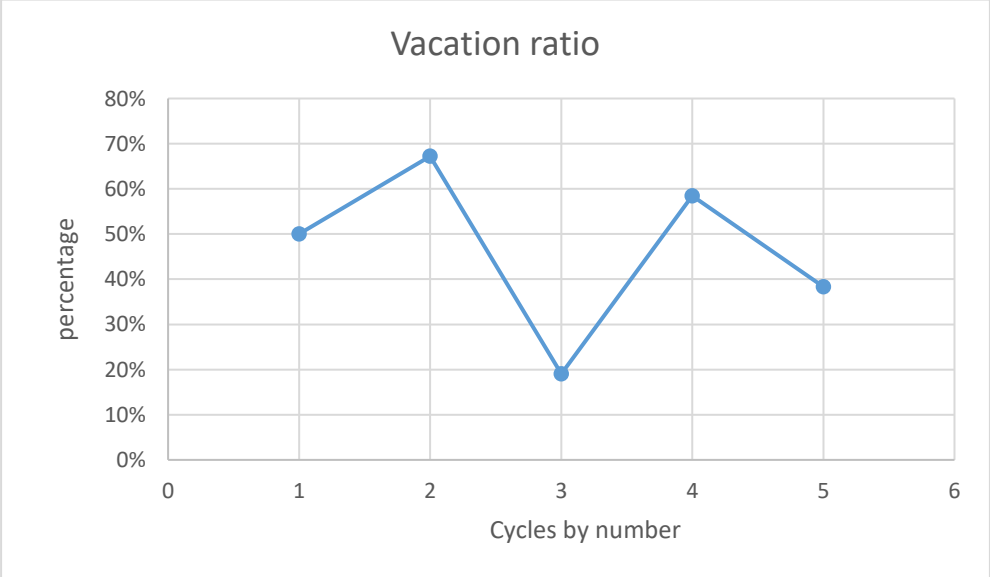

Figure 24: Working vacation in regenerative method

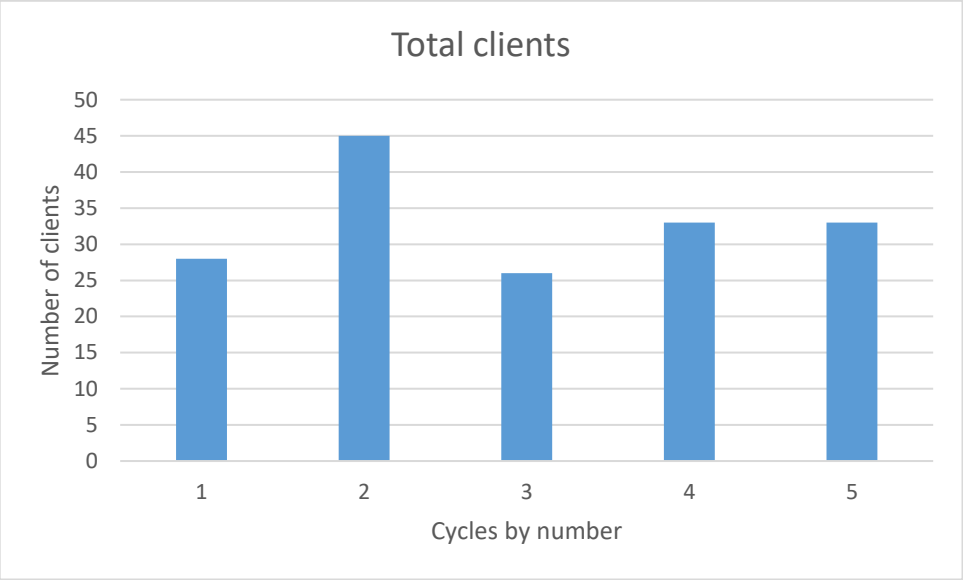

Figure 25: working vacation time

Here the table of statics

Table 2:Regenerative method result

Cycle	Max Queue	Average Queue	Total waiting	Vacation Duration	Vacation ratio	Total clients	Duration
1	6	1.82	91	25	50%	28	50
2	12	3.803278689	464	82	67%	45	122
3	13	6.421875	411	12	19%	26	64
4	8	3.285714286	253	45	58%	33	77
5	14	6.643835616	485	28	38%	33	73

4.6. The replication method

Its policy is to estimate performance measures by running multiple independent replications of the system, which means lunch serval simulation using different parameters each time, for that we changed the Arrival rate, Service time, the number of clients need served before passing to the preparation phase, and K the max queue size to keep the queue in working vacation.

We used low to high arrival rates while the other parameters are random

The parameter used in each simulation are in this table:

Table 3: Simulation parameters

Simulation	1	2	3	4	5
Arrival rate	0.5	0.5	0.75	0.9	0.25
Service time	5	8	10	4	6
Clients	10	20	30	50	8
K	5	10	8	20	10

The result shows that the queue size was proportional with the arrival rate, as it peaks in the simulation that used the highest arrival rate, and reach lowest average with the lowest rate excluding the first simulation that was used as witness as it has the same parameter with simulation used in the subinterval and regenerative method.

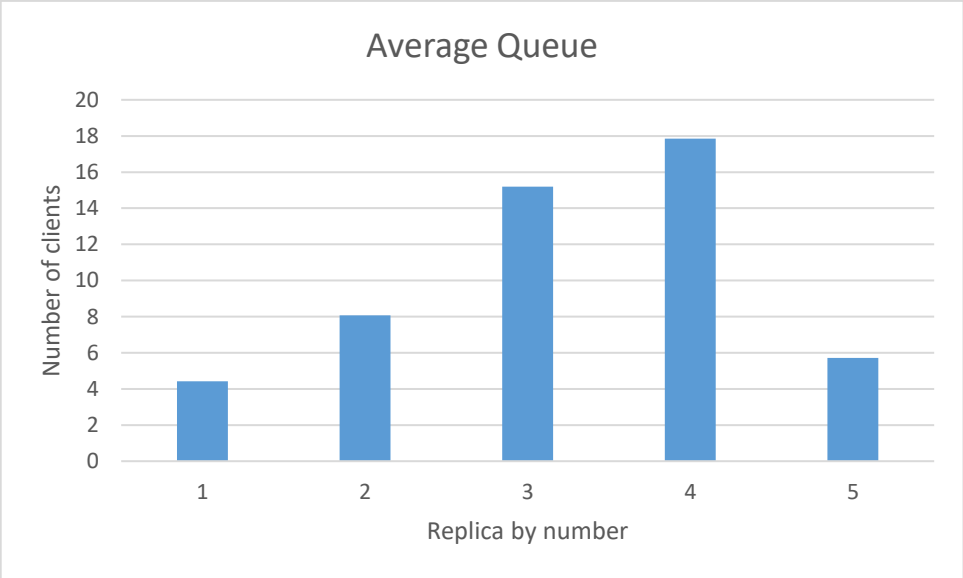

Figure 26: Replication method queue size

The queue ratio result was interesting, as it was lower than the witness test.

The third simulation had the lowest working vacation time because the high arrival rate overwhelms K size, making this set up not ideal for the system in which the secondary task is highly important .

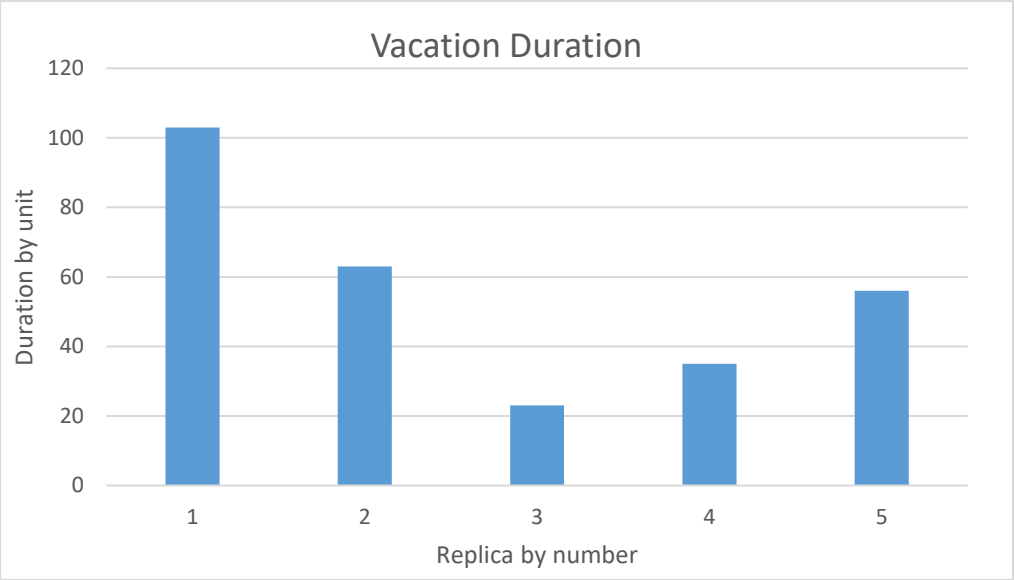

Figure 27:Workin vacation duration in Replication method

The table below shows a gigantic max queue size in test 3 and test 4 which also explains the little duration of working vacation and shows the need of this systems to increase the service rate of its servers.

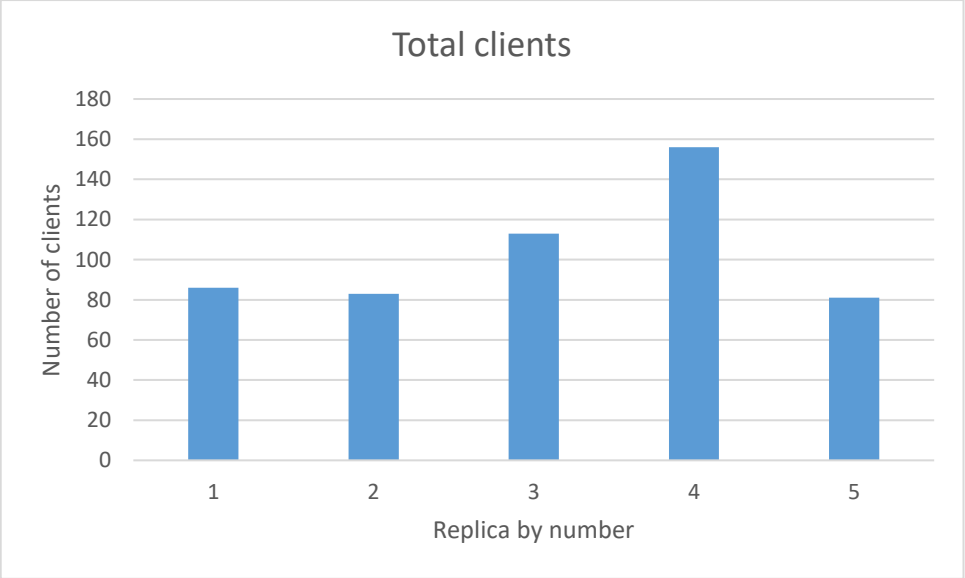

Table 4: Replication method result

Test	Max Queue	Average Queue	Total waiting	Vacation Duration	Vacation ratio	Total clients
1	14	4.42211055	888	103	52%	86
2	23	8.06532663	1605	63	32%	83
3	36	15.1909548	3036	23	12%	113
4	53	17.8442211	3604	35	18%	156
5	13	5.70854271	1140	56	28%	81

As a result of these 3 methods, we conclude that the system depends mostly at the parameters of the queue and the system doesn't achieve a steady stat, however it clear that queue runs in repeated stages of descent and ascent proportional with its three phases with differences in static.

We can deduce that in order to gain the most performance depending on the need a study of the best parameters is required and a simulation can confirm the specification needed.

4.7. Vacation and Working Vacation

As final test, we will simulation the first queue tested in three model, Regular queue, Vacation queue and Working vacation queue, where the vacation queue will stop serving the client during vacation instead of lowering the rate, with:

Arrival rate: 0.5

Service time: 5

Gate: 10

K: 5

It is worth noting that the regular queue has most constant system with an incredibly waiting time of only **128** and a stable queue size, since it has no phases.

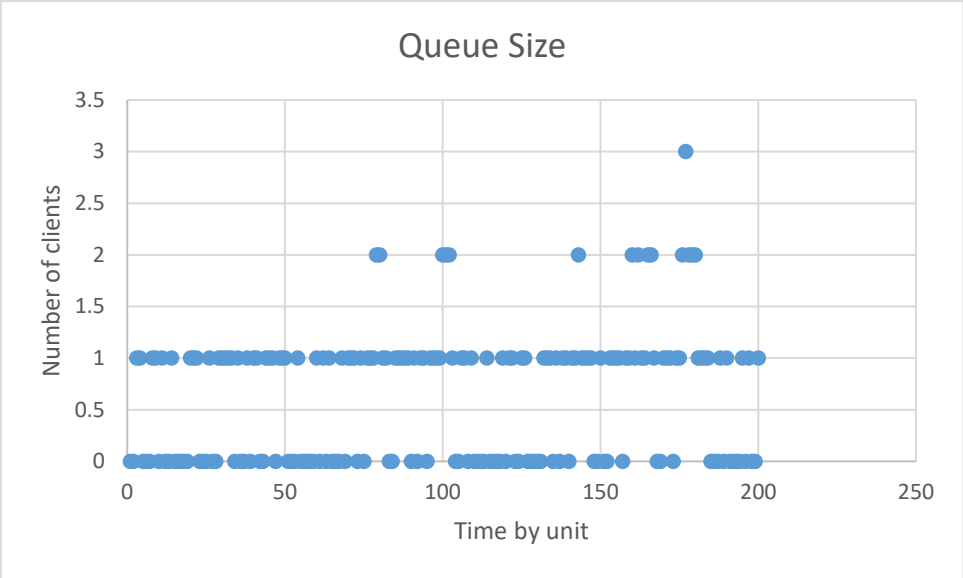

Figure 28: Queue size of regular queue

However, in system that require the servers to performs another task this model cannot be usable.

In other hand, the working vacation and vacation model show a similarity in client served but significantly the working vacation model shows a more constant performance while the vacation model pass by a lot of transition in such a small time, the servers switch to normal after spending short amount of time in vacation, and overall, 27% of the time in vacation while the WV model had 52%.

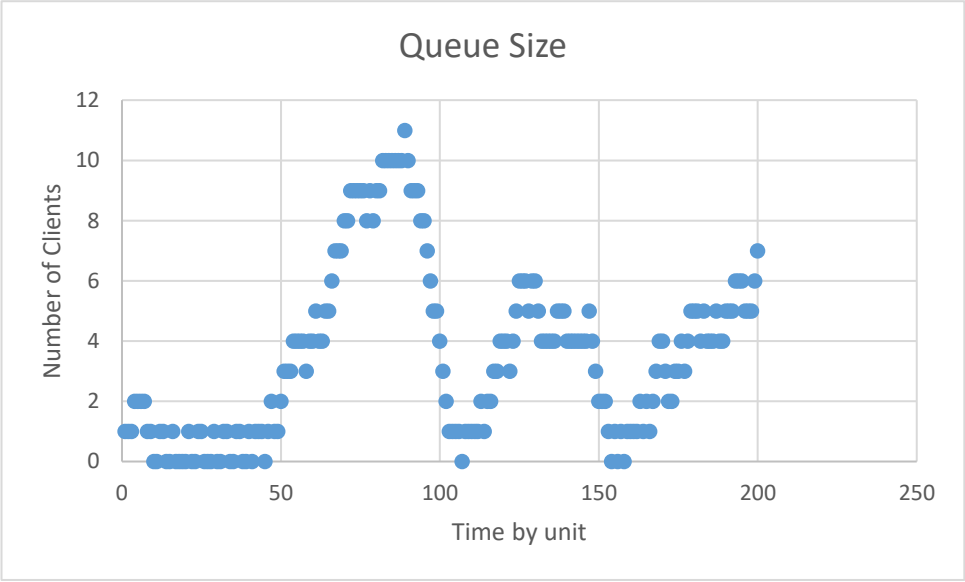

Figure 29: Working vacation model queue size

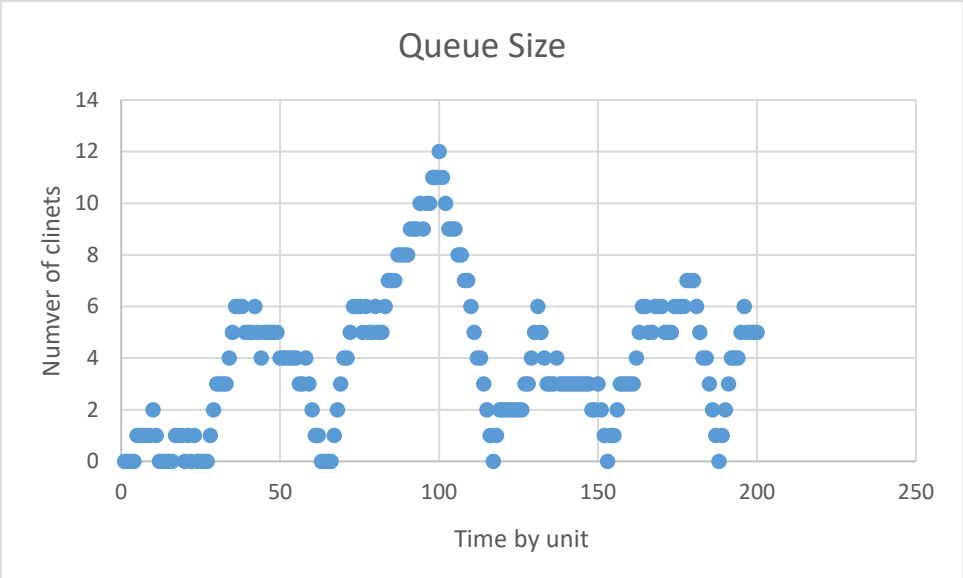

Figure 30: Vacation queue size

Here is the result of the 3 queue models

Table 5: Result of the simulation of the 3-queueing models

	Average queue size	Vacation ratio	Waiting time	Total client
Regular	0.64	0	128	95
WV	1.82	52%	780	89
Vacation	3.945	27%	789	88

As a result, the working vacation queue offers distinct advantages in terms of system continuity and service flexibility when compared to the vacation queue, particularly for systems that require seamless execution of multiple tasks without interrupting the main task. The working vacation approach ensures the system remains operational during idle periods by serving incoming customers at a reduced rate, thus maintaining a steady flow of tasks. This continuity is especially beneficial for applications that demand consistent service availability and cannot afford prolonged shutdowns or complete halts. Additionally, the flexibility of the working vacation method enables the system to adapt to varying workloads and demand patterns effectively, making it well-suited for dynamic environments where task requirements fluctuate over time. In contrast, the traditional vacation queue may encounter disruptions during vacation periods, leading to potential delays and service inconsistencies. Consequently, the working vacation strategy proves to be a valuable solution for enhancing system performance, reliability, and adaptability in scenarios that demand continuous and versatile task handling.

5. Conclusion

In this chapter we have focused on presenting the interesting results that have been obtained for our Simulator of Performance Analysis of Queueing Systems with Working Vacation. Furthermore, we have tried to analyse the obtained results by providing charts and tables that summarizes them in different use case scenarios.

General Conclusion

1. Discussion

In summary, the thesis on evaluating the performance of queuing systems with working vacation has successfully achieved its intended purpose by modelling and simulating queuing systems with working vacation. In this work, a simulation tool has been developed. It can collect power measurements using methods such as subinterval regeneration and replication techniques.

The research findings from the simulation experiments shed light on the advantages and drawbacks of the working vacation queue system compared to the traditional vacation queue. It is likely that the working vacation system demonstrated improved performance in certain scenarios due to its ability to handle low-intensity work periods effectively.

The thesis contributes valuable insights into the design and evaluation of queue systems with working vacation policies, offering a useful tool for researchers and practitioners in understanding and optimizing such systems. It highlights the importance of considering different operational scenarios and patterns when choosing between the traditional vacation and working vacation queue models.

2. Outlooks

Of course, a room of improvement is always opened, the tool still lacking a GUI (Graphic interface user) which will provide a visual interface that improves the user experience, making it easier for users to interact with the simulation tool. The GUI allows users to input parameters, set up simulation scenarios, and view results in a more intuitive and accessible manner, moreover real-time visualization through graphs and charts enables users to observe the behaviour of the queue system during simulation, gaining immediate insights into its performance dynamics without using intermediate programs, in addition, since the subject is based on a chaotic process, multiple tests are recommended to verify the results of the simulation.

This research provides a foundation for further investigations and enhancements in queueing systems with working vacation, as well as encourages the exploration of other innovative queue management techniques for optimizing system performance in various real-world applications.

Bibliography

Bibliography

[1] William K. Sewart: Probability, Markov chains, Queues, and Simulation

[2] Naishuo, T., & Zhang, Z. G. (2006). Vacation Queueing Models: Theory and Applications

[3] J J O'Connor and E F Robertson University of St Andrews, Scotland

[4] Fadhil M. Al- Rubaee,Saad Talib Hasson Simulation approach to model queuing Problems

[5] Jerry Banks,John S. Carson II,Barry L Nelson,David M. Nicol .Discrete-Event System Simulation

[6] Manuele Leonelli. Simulation and Modelling to Understand Change. 2021-04-19

[7] J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds. INTRODUCTION TO MONTE CARLO SIMULATION

[8] M. Kias Mohamed, M. Nekaa Hamza, Un Outil de Simulation pour l'analyse des Performances d'un Noeud Capteur en Utilisant les Politiques de Vacances

[9] KHALAF, Rehab, "On some queueing systems with server vacations, extended vacations, breakdowns, delayed repairs and stand-bys.," 2012. Thèse de doctorat. Brunel University, School of Information Systems, Computing and Mathematics..

[10] Solaiman, B. (2006). Processus stochastiques pour l'ingénieur. PPUR presses polytechniques.

[11] Dong-Yuh Yang, Chia-Huang Wu, Cost-minimization analysis of a working vacation queue with N-policy and server breakdowns

[12] An introduction to Markov chains and their applications within finance

[13] M. Kias Mohamed, M. Nekaa Hamza, Un Outil de Simulation pour l'analyse des Performances d'un Noeud Capteur en Utilisant les Politiques de Vacance.

[14] Mar 23, 2023 | Performance Testing, Tech Tips / https://www.loadview-testing.com/blog/performance-testing-baseline-and-benchmark-testing/

[15] https://thefunctionalba.com/2019/11/how-to-analyze-performance-measures/

[16] Performance Monitoring and Evaluation using direct observation techniques

[17] Indian Institute of Technology Kanpur http://home.iitk.ac.in/~skb/ee679/chapter_7.html

[18] [ACM Computing SurveysVolume 18Issue 101 March 1986pp 39–65https://doi.org/10.1145/6462.6485]

[19] Mian Zhang, Zhengting Hou, Performance analysis of M/G/1 queue with working-vacations and vacation interruption

[20] Amtout Djedjiga, Mallek Saliha, Application des modèles avec vacances